Introduction to Grammars

Mattox Beckman

beckman@iit.edu

Illinois Institute of Technology

Your goal for this lecture is to learn how to do the following things:

- Identify and explain the parts of a grammar.
- Define *terminal*, *nonterminal*, *production*, *sentence*, *parse tree*, *left-recursive*, *ambiguous*.
- Use a grammar to draw the parse tree of a sentence.
- Identify a grammar that is *left-recursive*.
- Know about *ambiguous grammars*:
  - Be able to identify, demonstrate, and eliminate ambiguity.
- Define *First Set* and *Follow Set*.
- Compute the First Set from a grammar.
- Compute the Follow Set from a grammar.

Reminder: The Problem

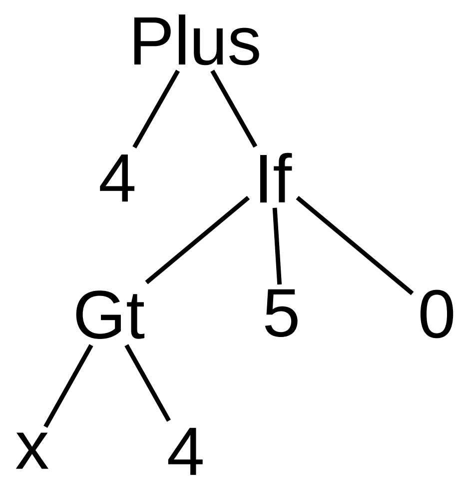
- Computer programs are entered as a stream of ASCII characters.

```
4 + if x > 4 then 5 else 0
```

- We want to convert them into an *Abstract Syntax Tree*

ML code

```
PlusExp(
  IntExp 4,
  IfExp(
    GtExp(VarExp "x",
          IntExp 4),
    IntExp 5,
    IntExp 0))
```



Reminder: The Solution

Characters $\longrightarrow$ $\boxed{\text{Lexer}}$ $\longrightarrow$ Tokens $\longrightarrow$ $\boxed{\text{Parser}}$ $\longrightarrow$ Tree

The conversion from strings to trees is accomplished in two steps.

- First, convert the stream of characters into a stream of *tokens*.
  - This is called *lexing* or *scanning*.
  - Turns characters into words and categorizes them.
  - We did this in the last two lectures!
- Second, convert the stream of tokens into an abstract syntax tree.
  - This is called *parsing*.
  - Turns words into *sentences*.

What is in a sentence?

When we specify a sentence, we talk about two things that could be in them.

1. *Terminals*: tokens that are atomic — they have no smaller parts (e.g., “nouns”, “verbs”, “articles”)
2. *Non-terminals*: clauses that are not atomic — they are broken into smaller parts (e.g. “prepositional phrase”, “independent clause”, “predicate”)

Examples: (identify the terminals and the non-terminals)

- A sentence is a noun phrase, a verb, and a prepositional phrase
- A noun phrase is a determinant, and a noun
- A prepositional phrase is a preposition and a noun phrase.

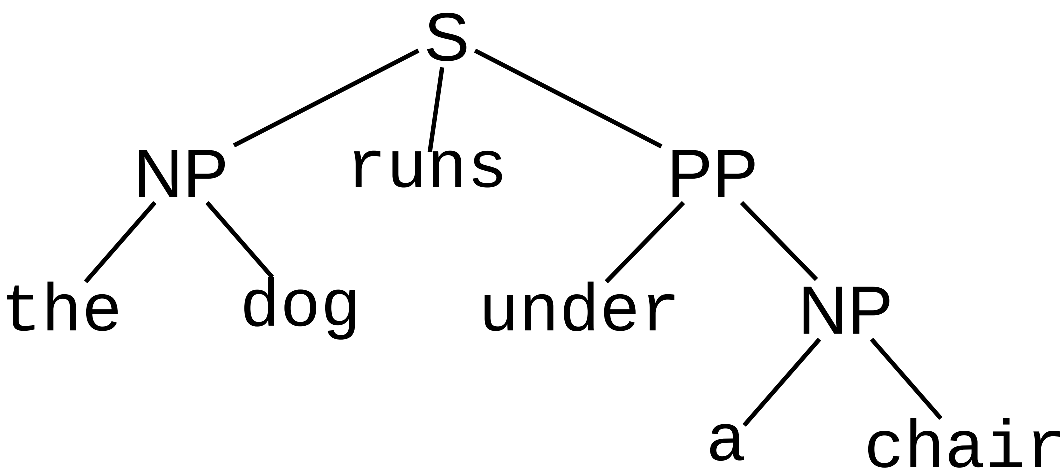
$$S \rightarrow \text{N verb P}$$
$$N \rightarrow \text{det noun}$$
$$P \rightarrow \text{prep N}$$

- Each of the above lines is called a *production*.
  The *symbol* on the left hand side can be *produced* by collecting the symbols on the right hand side.
- The capital identifiers are *non-terminal* symbols.
- The lower case identifiers are *terminal* symbols.
- Because the left hand side is only a single non-terminal, the rules are *context free*. (Contrast: x S → NP verb PP)

Grammars specify trees...

"The dog runs under a chair."

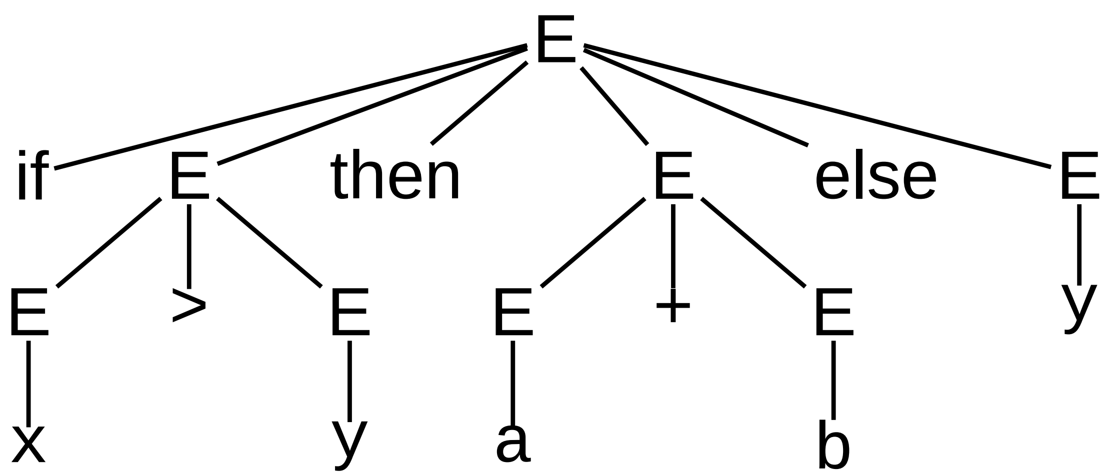
$$\text{S} \rightarrow \text{NP verb PP}$$
$$\text{NP} \rightarrow \text{det noun}$$
$$\text{PP} \rightarrow \text{prep NP}$$



Another Example...

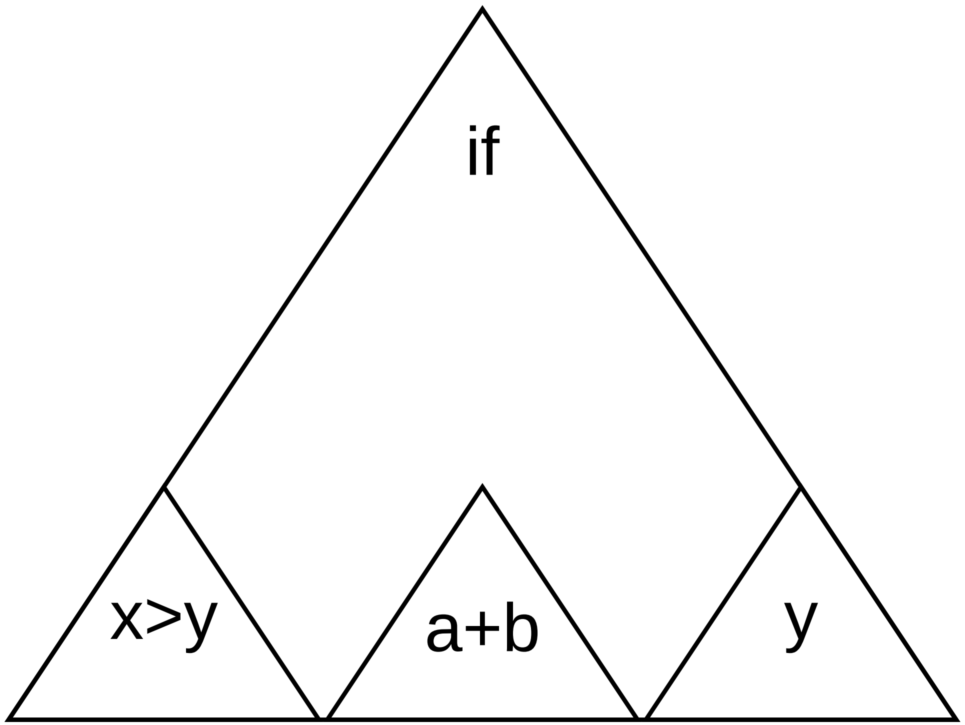
$$
\begin{aligned}
E &\longrightarrow E + E \\
&\longrightarrow v \\
&\longrightarrow E > E \\
&\longrightarrow \text{if } E \text{ then } E \text{ else } E
\end{aligned}
$$

`if x > y then a + b else y`



Triangle Form

```
if x > y then a + b else y
```



It is important to be able to say what properties a grammar has.

Epsilon Productions A production of the form "E $\rightarrow \epsilon$", where ϵ represents the empty string.

Right Linear Grammars where all the productions have the form "E $\rightarrow$ x F" or "E $\rightarrow$ x".

Left-Recursive a production like "E $\rightarrow$ E + X"

Ambiguous More than one parse tree is possible for a specific sentence.

Epsilon Productions

- Sometimes we want to specify that a symbol can become nothing.
- Example: "E $\rightarrow \epsilon$"
- Another example:

  S $\rightarrow$ NP verb PP
  NP $\rightarrow$ det A noun
  PP $\rightarrow$ prep NP
  A $\rightarrow$ adjective A
  A $\rightarrow \epsilon$

  This says that adjectives are an optional part of noun phrases.

- A *right linear* grammars is one in which all the productions have the form
  "E $\rightarrow$ x A" or "E $\rightarrow$ x".
- This corresponds to the *regular languages*.
- Example: regular expression `(10)*23` describes same language as this grammar:

$$A_0 \rightarrow 1A_1 \mid 2A_2$$

$$A_1 \rightarrow 0A_0$$

$$A_2 \rightarrow 3A_3$$

$$A_3 \rightarrow \epsilon$$

- A grammar is *recursive* if the symbol being produced (the one on the left-hand side) also appears in the right hand side.

  Example: "E $\rightarrow$ if E then E else E"

- A grammar is *left-recursive* if the production symbol appears as the first symbol on the right-hand-side.

  Example: "E $\rightarrow$ E + F"

- ... or if is produced by a chain of left recursions ...

  Example:
  $$A \rightarrow B\text{x}$$
  $$B \rightarrow A\text{y}$$

- Question: why do we care if it's left-recursive?

- One way to use a grammar is to form a *recursive descent parser*.
- Start with a grammar and make some types ...

  S $\longrightarrow$ NP verb PP

  NP$\longrightarrow$ det noun

  PP$\longrightarrow$ prep NP

```
type tree = S of (tree * token * tree)
          | NP of (token * token)
          | PP of (token * tree)
and token = Verb of string
          | Det of string
          | Noun of string
          | Prep of string
```

- Next, write a function for each of the productions.
- Each function eats some terminal symbols and returns a pair...

```
let getS tlist =
  let (np, newlist) = getNP tlist in
  let (verb, newlist) = getVerb newlist in
  let (pp, newlist) = getPP newlist in
    S (np,verb,pp), newlist
and getVerb tlist = match (hd tlist) with
  Verb x -> (Verb x, tl tlist)
| _ -> raise (Failure "parse error")
and getNP tlist = ...
```

- What would happen if we had a left-recursive rule?

Ambiguous Grammars

- A grammar is *ambiguous* if it can produce more than one parse tree for a single sentence.
- There are two common forms of ambiguity:
  - The “dangling else” form:

    $E \longrightarrow$ if E then E else E

    $E \longrightarrow$ if E then E

    $E \longrightarrow$ whatever

    Example: `if a then if x then y else z` ... to which `if` does the `else` belong?
  - The “double-ended recursion” form: $E \longrightarrow E + E$, $E \longrightarrow E * E$

    Example “3 + 4 * 5” ... is it “(3 + 4) * 5” or “3 + (4 * 5)”?

Fixing Ambiguity

- Ambiguity can often be eliminated by thinking more carefully about what you are trying to express with your grammar.
- “Dangling else” usually matches with the nearest `if`. This can be encoded in the grammar. See §4.3 of the Dragon Book for details.

- The "double-ended recursion" form usually reveals a lack of precedence and associativity information. A technique called *stratification* often fixes this.
  - Left-recursive means "associates to the left", similarly right-recursive.
  - Higher precedence rules occur lower in the grammar.

E $\longrightarrow$ F + E

E $\longrightarrow$ F

F $\longrightarrow$ T * E

F $\longrightarrow$ T

T $\longrightarrow$ (E)

T $\longrightarrow$ integer

Try the following problems

1. Draw two separate parse trees for the sentence `2 + 3 * 5` given the grammar

   $E \rightarrow E + E$

   $E \rightarrow E * E$

   $E \rightarrow$ integer

2. Now draw a tree for the same sentence using the grammar

$E \rightarrow F + E$	$F \rightarrow T * E$	$T \rightarrow (E)$
$E \rightarrow F$	$F \rightarrow T$	$T \rightarrow$ integer

3. Draw two separate parse trees for the “dangling else” example:

   `if a then if x then y else z`

   $E \rightarrow$ if E then E else E

   $E \rightarrow$ if E then E

   $E \rightarrow$ var

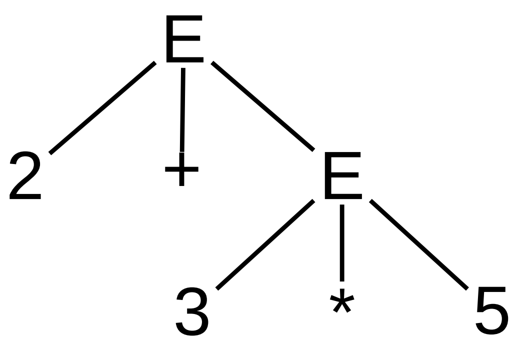
Problem 1 §3 **Activity!**

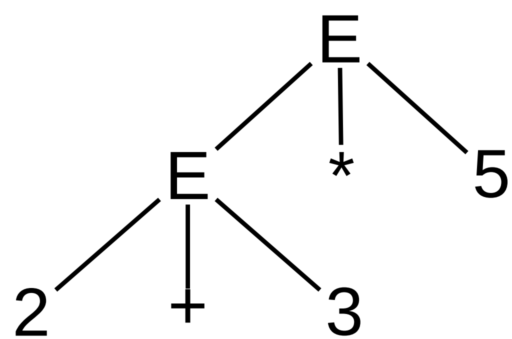
- Draw two separate parse trees for the sentence `2 + 3 * 5` given the grammar

  $E \rightarrow E + E$

  $E \rightarrow E * E$

  $E \rightarrow$ integer





Problem 2

- Now draw a tree for the same sentence using the grammar

  $E \longrightarrow F + E$

  $E \longrightarrow F$

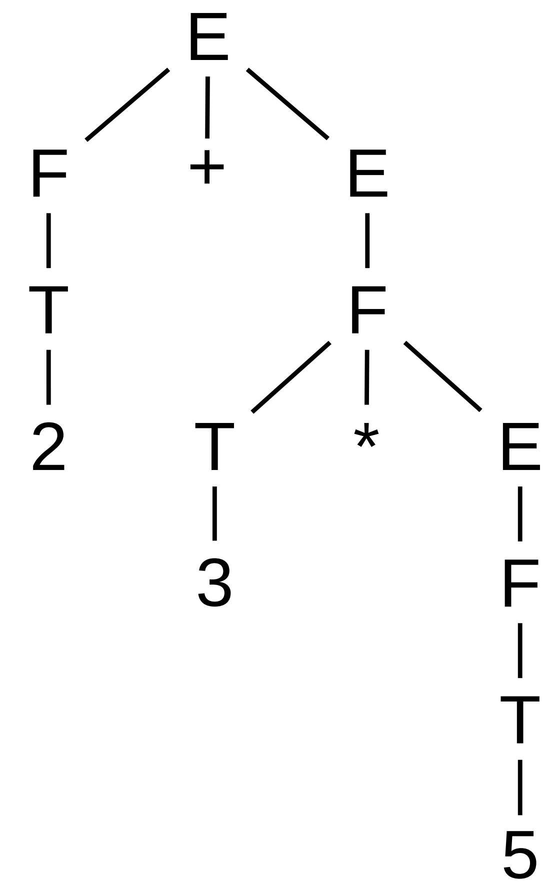
  $F \longrightarrow T * E$

  $F \longrightarrow T$

  $T \longrightarrow (E)$

  $T \longrightarrow$ integer

These large tree sizes can cause our parser to be inefficient.

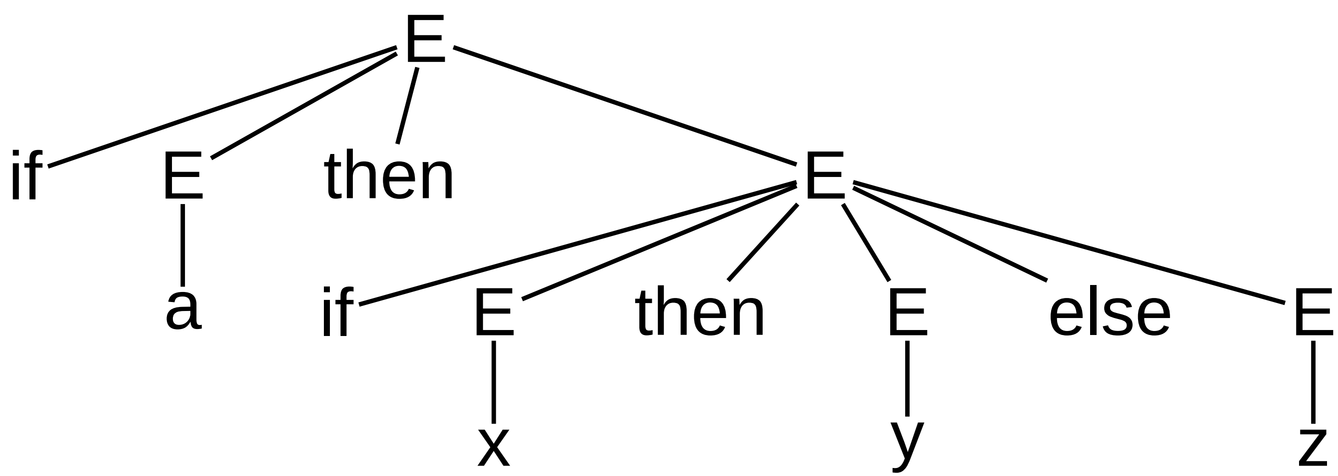


- Draw two separate parse trees for the "dangling else" example:

`if a then if x then y else z`

E→ if E then E else E
E→ if E then E
E→ var

Tree 1:



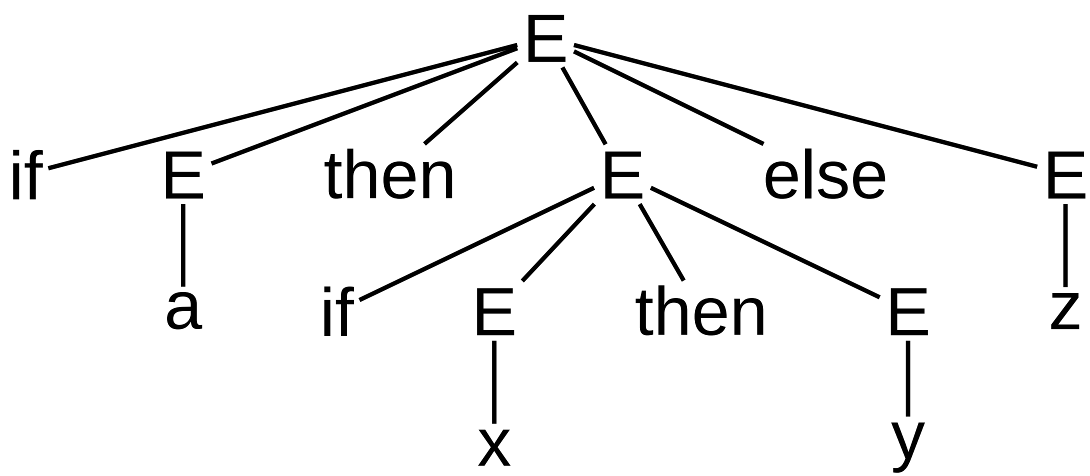
Problem 3b

- Draw two separate parse trees for the "dangling else" example:

`if a then if x then y else z`

E$\longrightarrow$ if E then E else E
E$\longrightarrow$ if E then E
E$\longrightarrow$ var

Tree 2:



- Given a grammar for a language L, how can we recognize a sentence in L?
- Solution: Divide and Conquer: Given a symbol E...
  - What symbols indicate that the symbol E is just starting? (First Set)
  - What symbols should we expect to see after we have finished parsing an E?

Misleadingly simple example:

$$S \rightarrow xEy \qquad \text{First}(E) = \{z, q\}$$
$$E \rightarrow zE \qquad \text{Follow}(E) = \{y\}$$
$$E \rightarrow q$$

- Important because a parser can see only a few tokens at once.

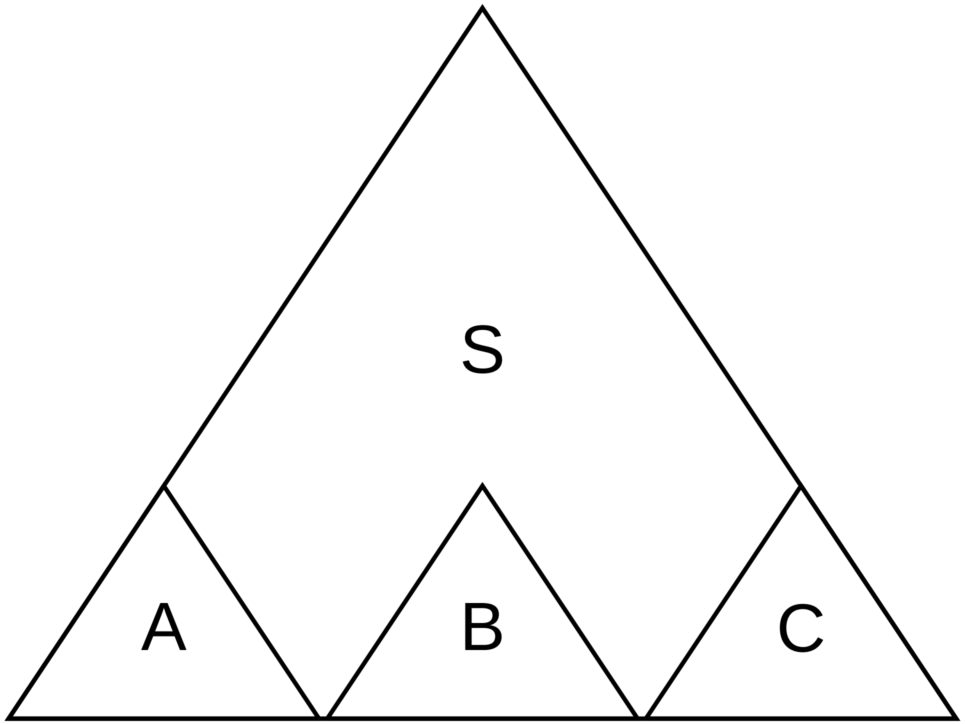
Algorithm

We can compute the FIRST set by a simple iterative algorithm.
For each symbol X.

1. if X is a terminal, then $First(X) = \{X\}$
2. if there is a production $X \rightarrow \epsilon$, then add ϵ to $First(X)$.
3. if there is a production $X \rightarrow Y_1 Y_2 \cdots Y_n$, then add $First(Y_1 Y_2 \cdots Y_n)$ to $First(X)$:
   - If $First(Y_1)$ does not contain ϵ, then $First(Y_1 Y_2 \cdots Y_n) = First(Y_1)$.
   - Otherwise, $First(Y_1 Y_2 \cdots Y_n) = First(Y_1)/\epsilon \cup First(Y_2 \cdots Y_n)$
   - If all of $Y_1, Y_2, \ldots Y_n$ have ϵ then add ϵ to $First(X)$.

First Set Picture

$$S \rightarrow \; A \; B \; C$$



First Set Example

Consider the grammar
$$S \longrightarrow \texttt{if}\ E\ \texttt{then}\ S\ ;$$
$$S \longrightarrow \texttt{print}\ E;$$
$$E \longrightarrow E\ \texttt{+}\ E$$
$$E \longrightarrow P\ \texttt{id}$$
$$P \longrightarrow *\ P$$
$$P \longrightarrow \epsilon$$

Step 1: Create a list of symbols....

S={}

E={}

P={}

Consider the grammar

$$S \longrightarrow \texttt{if}\ E\ \texttt{then}\ S\ ; \Leftarrow$$
$$S \longrightarrow \texttt{print}\ E; \Leftarrow$$
$$E \longrightarrow E\ \texttt{+}\ E$$
$$E \longrightarrow P\ \texttt{id}$$
$$P \longrightarrow *\ P \Leftarrow$$
$$P \longrightarrow \epsilon \Leftarrow$$

Step 2: Add terminals starting productions, and all ϵ.

S={`if`, `print` }

E={}

P={ϵ, `*`}

Consider the grammar

$$S \longrightarrow \texttt{if}\ E\ \texttt{then}\ S\ ;$$
$$S \longrightarrow \texttt{print}\ E;$$
$$E \longrightarrow E\ \texttt{+}\ E$$
$$E \longrightarrow P\ \texttt{id} \Leftarrow$$
$$P \longrightarrow *\ P$$
$$P \longrightarrow \epsilon$$

Step 3: Check productions. Add $First(P\texttt{id})$ to $First(E)$.

S={`if`, `print` }
E={`*`, `id`}
P={ϵ, `*`}

Consider the grammar

$$S \longrightarrow \texttt{if}\ E\ \texttt{then}\ S\ ;$$
$$S \longrightarrow \texttt{print}\ E;$$
$$E \longrightarrow E\ \textbf{+}\ E \Leftarrow$$
$$E \longrightarrow P\ \texttt{id}$$
$$P \longrightarrow *\ P$$
$$P \longrightarrow \epsilon$$

Step 3: Check productions: $E \longrightarrow E + E$ adds nothing.

S={`if`, `print` }

E={`*`, `id`}

P={ϵ, `*`}

Another First Set Example

$$S \to A\mathtt{x}$$
$$S \to B\mathtt{y}$$
$$S \to \mathtt{z}$$
$$A \to \mathtt{1}CB$$
$$A \to \mathtt{2}B$$
$$B \to \mathtt{3}B$$
$$B \to C$$
$$C \to \mathtt{4}$$
$$C \to \epsilon$$

Create a chart:

S={}

A={}

B={}

C={}

Another First Set Example

$S \rightarrow A\mathtt{x}$

$S \rightarrow B\mathtt{y}$

$S \rightarrow \mathtt{z} \Leftarrow$

$A \rightarrow \mathtt{1}CB \Leftarrow$

$A \rightarrow \mathtt{2}B \Leftarrow$

$B \rightarrow \mathtt{3}B \Leftarrow$

$B \rightarrow C$

$C \rightarrow \mathtt{4} \Leftarrow$

$C \rightarrow \epsilon \Leftarrow$

Add initial terminals and ϵs.

S={z}

A={1, 2}

B={3}

C={ϵ, 4}

Another First Set Example

$S \rightarrow A\mathtt{x} \Leftarrow$

$S \rightarrow B\mathtt{y}$

$S \rightarrow \mathtt{z}$

$A \rightarrow \mathtt{1}CB$

$A \rightarrow \mathtt{2}B$

$B \rightarrow \mathtt{3}B$

$B \rightarrow C$

$C \rightarrow \mathtt{4}$

$C \rightarrow \epsilon$

Add $First(A\mathtt{x})$ to $First(S)$.

S={z, 1, 2}

A={1, 2}

B={3}

C={ϵ, 4}

Another First Set Example

$$S \to A\mathtt{x}$$
$$S \to B\mathtt{y} \Leftarrow$$
$$S \to \mathtt{z}$$
$$A \to \mathtt{1}CB$$
$$A \to \mathtt{2}B$$
$$B \to \mathtt{3}B$$
$$B \to C$$
$$C \to \mathtt{4}$$
$$C \to \epsilon$$

Add $First(B\text{y})$ to $First(S)$.

S={z, 1, 2, 3}

A={1, 2}

B={3}

C={ϵ, 4}

Note that there is still more to be added to $First(B)$! We will have to revisit this step later.

Another First Set Example

$$S \to A\mathtt{x}$$
$$S \to B\mathtt{y}$$
$$S \to \mathtt{z}$$
$$A \to \mathtt{1}CB$$
$$A \to \mathtt{2}B$$
$$B \to \mathtt{3}B$$
$$B \to C \Leftarrow$$
$$C \to \mathtt{4}$$
$$C \to \epsilon$$

Add $First(C)$ to $First(B)$.

S={z, 1, 2, 3}

A={1, 2}

B={3, 4, ϵ}

C={ϵ, 4}

At this point we should iterate again to see if anything changes.

$S \rightarrow A\texttt{x} \Leftarrow$
$S \rightarrow B\texttt{y}$
$S \rightarrow \texttt{z}$
$A \rightarrow \texttt{1}CB$
$A \rightarrow \texttt{2}B$
$B \rightarrow \texttt{3}B$
$B \rightarrow C$
$C \rightarrow \texttt{4}$
$C \rightarrow \epsilon$

Add $First(A\text{x})$ to $First(S)$ again.

S={z, 1, 2, 3}
A={1, 2}
B={3, 4, ϵ }
C={ϵ, 4}

Nothing happens...

Another First Set Example

$S \rightarrow A\texttt{x}$

$S \rightarrow B\texttt{y} \Leftarrow$

$S \rightarrow \texttt{z}$

$A \rightarrow \texttt{1}CB$

$A \rightarrow \texttt{2}B$

$B \rightarrow \texttt{3}B$

$B \rightarrow C$

$C \rightarrow \texttt{4}$

$C \rightarrow \epsilon$

Add $First(B\text{y})$ to $First(S)$ again.

S={z, 1, 2, 3, 4, y}

A={1, 2}

B={3, 4, ϵ }

C={ϵ, 4}

The 4 gets propagated.... also, since B could be ϵ we need to add y.

Another First Set Example

$S \to A\mathtt{x}$

$S \to B\mathtt{y}$

$S \to \mathtt{z}$

$A \to \mathtt{1}CB$

$A \to \mathtt{2}B$

$B \to \mathtt{3}B$

$B \to C \Leftarrow$

$C \to \mathtt{4}$

$C \to \epsilon$

Add $First(C)$ to $First(B)$ again.

S={z, 1, 2, 3, 4, y}

A={1, 2}

B={3, 4, ϵ }

C={ϵ, 4}

Nothing happens...

At this point, if we should iterate again. But nothing happens when you do that, so we are done.

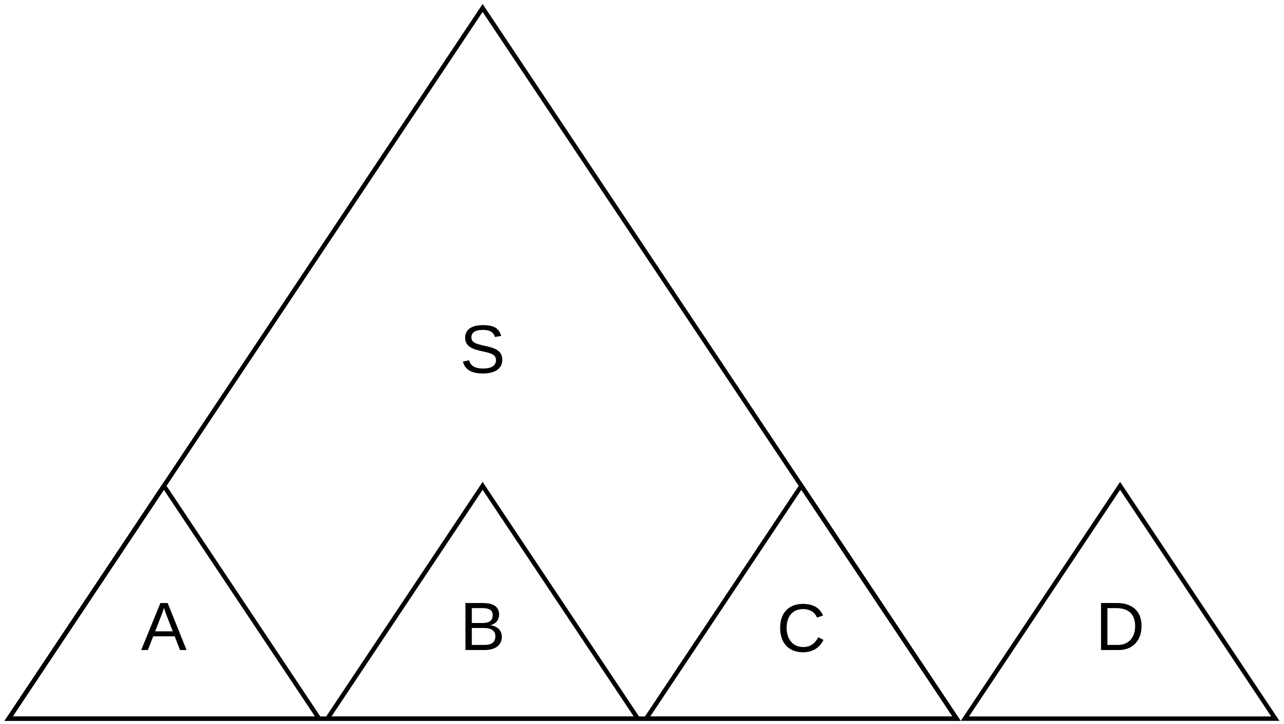
- Given a non-terminal symbol S, what terminal symbols could come after strings that are derived from S?

The Algorithm:

1. Put \$ in $FOLLOW(S)$, where S is the start symbol.
   \$ represents the "end of input."
2. If there is a production $X \rightarrow \alpha Y \beta$, then add $FIRST(\beta)$ (but not ϵ) to $FOLLOW(Y)$.
3. If there is a production $X \rightarrow \alpha Y$, or if there is a production $X \rightarrow \alpha Y \beta$, where $\epsilon \in FIRST(\beta)$ then add $FOLLOW(X)$ to $FOLLOW(Y)$.

Follow Set Picture

$S \rightarrow \; A \; B \; C,$ and let D in Follow(S)



Follow Set Example

$$S \longrightarrow \texttt{if}\ E\ \texttt{then}\ S\ ;$$
$$S \longrightarrow \texttt{print}\ E;$$
$$E \longrightarrow E\ \texttt{+}\ E$$
$$E \longrightarrow P\ \texttt{id}\ P$$
$$P \longrightarrow *\ P$$
$$P \longrightarrow \epsilon$$

Make a chart, add $ to S.

S={$}
E={}
P={}

$S \longrightarrow \texttt{if}\ E\ \texttt{then}\ S\ ;\ \Leftarrow$

$S \longrightarrow \texttt{print}\ E;$

$E \longrightarrow E\ \texttt{+}\ E$

$E \longrightarrow P\ \texttt{id}\ P$

$P \longrightarrow *\ P$

$P \longrightarrow \epsilon$

Check productions: add `then` to $FOLLOW(E)$,
and ; to $FOLLOW(S)$

S={\$, ;}

E={then}

P={}

Follow Set Example

$S \longrightarrow \texttt{if}\ E\ \texttt{then}\ S\ ;$

$S \longrightarrow \texttt{print}\ E; \Leftarrow$

$E \longrightarrow E + E \Leftarrow$

$E \longrightarrow P\ \texttt{id}\ P$

$P \longrightarrow * P$

$P \longrightarrow \epsilon$

Check productions: add ; and + to $FOLLOW(E)$

S={\$, ;}

E={then, ;, +}

P={}

Follow Set Example

$S \rightarrow \texttt{if}\ E\ \texttt{then}\ S\ ;$

$S \rightarrow \texttt{print}\ E;$

$E \rightarrow E + E$

$E \rightarrow P\ \texttt{id}\ P \Leftarrow$

$P \rightarrow * P$

$P \rightarrow \epsilon$

Check productions: add `id` to $FOLLOW(P)$

S={$, ;}

E={`then`, ;, +}

P={`id`}

$S \longrightarrow \texttt{if}\ E\ \texttt{then}\ S\ ;$

$S \longrightarrow \texttt{print}\ E;$

$E \longrightarrow E + E$

$E \longrightarrow P\ \texttt{id}\ P \Leftarrow$

$P \longrightarrow *\ P$

$P \longrightarrow \epsilon$

Check endings: P ends this rule, so add $FOLLOW(E)$ to $FOLLOW(P)$.

S={$, ;}

E={`then`, ;, +}

P={`id`, `then`, ;, +}

$$S \rightarrow \texttt{if}\ E\ \texttt{then}\ S\ ;$$
$$S \rightarrow \texttt{print}\ E;$$
$$E \rightarrow E\ \texttt{+}\ E$$
$$E \rightarrow P\ \texttt{id}\ P$$
$$P \rightarrow *\ P$$
$$P \rightarrow \epsilon$$

Done.

S={$, ;}

E={`then`, ;, +}

P={`id`, `then`, ;, +}

Another Follow Set Example

$$S \rightarrow A\mathtt{x}$$
$$S \rightarrow B\mathtt{y}$$
$$S \rightarrow \mathtt{z}$$
$$A \rightarrow \mathtt{1}CB$$
$$A \rightarrow \mathtt{2}B$$
$$B \rightarrow \mathtt{3}B$$
$$B \rightarrow C$$
$$C \rightarrow \mathtt{4}$$
$$C \rightarrow \epsilon$$

S={$}
A={}
B={}
C={}

Create a table, and add $ to $Follow(S)$.

Another Follow Set Example

$S \rightarrow A\mathtt{x} \Leftarrow$

$S \rightarrow B\mathtt{y}$

$S \rightarrow \mathtt{z}$

$A \rightarrow \mathtt{1}CB$

$A \rightarrow \mathtt{2}B$

$B \rightarrow \mathtt{3}B$

$B \rightarrow C$

$C \rightarrow \mathtt{4}$

$C \rightarrow \epsilon$

S={ $}

A={x}

B={}

C={}

Add x to $Follow(A)$.

Another Follow Set Example

$S \longrightarrow A\mathtt{x}$

$S \longrightarrow B\mathtt{y} \Leftarrow$

$S \longrightarrow \mathtt{z}$

$A \longrightarrow \mathtt{1}CB$

$A \longrightarrow \mathtt{2}B$

$B \longrightarrow \mathtt{3}B$

$B \longrightarrow C$

$C \longrightarrow \mathtt{4}$

$C \longrightarrow \epsilon$

S={ $}

A={x}

B={y}

C={}

Add y to $Follow(B)$.

Another Follow Set Example

$S \rightarrow A\mathtt{x}$

$S \rightarrow B\mathtt{y}$

$S \rightarrow \mathtt{z} \Leftarrow$

$A \rightarrow \mathtt{1}CB$

$A \rightarrow \mathtt{2}B$

$B \rightarrow \mathtt{3}B \Leftarrow$

$B \rightarrow C$

$C \rightarrow \mathtt{4} \Leftarrow$

$C \rightarrow \epsilon \Leftarrow$

S={ $}

A={x}

B={y}

C={}

These productions add nothing.

Another Follow Set Example

$S \rightarrow A\mathtt{x}$

$S \rightarrow B\mathtt{y}$

$S \rightarrow \mathtt{z}$

$A \rightarrow \mathtt{1}CB \Leftarrow$

$A \rightarrow \mathtt{2}B$

$B \rightarrow \mathtt{3}B$

$B \rightarrow C$

$C \rightarrow \mathtt{4}$

$C \rightarrow \epsilon$

S={ $}

A={x}

B={y}

C={3, 4}

Add $First(B)$ to $Follow(C)$

Another Follow Set Example

$S \rightarrow A\texttt{x}$

$S \rightarrow B\texttt{y}$

$S \rightarrow \texttt{z}$

$A \rightarrow \texttt{1}CB \Leftarrow$

$A \rightarrow \texttt{2}B \Leftarrow$

$B \rightarrow \texttt{3}B$

$B \rightarrow C$

$C \rightarrow \texttt{4}$

$C \rightarrow \epsilon$

S={ $}

A={x}

B={x, y}

C={3, 4}

Add $Follow(A)$ to $Follow(B)$.

Another Follow Set Example

$S \rightarrow A\mathtt{x}$

$S \rightarrow B\mathtt{y}$

$S \rightarrow \mathtt{z}$

$A \rightarrow \mathtt{1}CB \Leftarrow$

$A \rightarrow \mathtt{2}B$

$B \rightarrow \mathtt{3}B$

$B \rightarrow C$

$C \rightarrow \mathtt{4}$

$C \rightarrow \epsilon$

S={ $}

A={x}

B={x, y}

C={x, 3, 4}

B can become ϵ, so add $Follow(A)$ to $Follow(C)$.

Another Follow Set Example

$S \rightarrow A\mathtt{x}$

$S \rightarrow B\mathtt{y}$

$S \rightarrow \mathtt{z}$

$A \rightarrow \mathtt{1}CB$

$A \rightarrow \mathtt{2}B$

$B \rightarrow \mathtt{3}B$

$B \rightarrow C \Leftarrow$

$C \rightarrow \mathtt{4}$

$C \rightarrow \epsilon$

S={ $}

A={x}

B={x, y}

C={x, y, 3, 4}

Add $Follow(B)$ to $Follow(C)$. Now we're done.

Compute the First and Follow sets for this grammar.

$$E \rightarrow T\ E'$$
$$E' \rightarrow +\ T\ E'$$
$$E' \rightarrow \epsilon$$
$$T \rightarrow F\ T'$$
$$T' \rightarrow *\ F\ T'$$
$$T' \rightarrow \epsilon$$
$$F \rightarrow (\ E\ )$$
$$F \rightarrow \texttt{id}$$

Shorthand notation:

$$E \rightarrow T\ E'$$
$$E' \rightarrow +\ T\ E'\ |\epsilon$$
$$T \rightarrow F\ T'$$
$$T' \rightarrow *\ F\ T'\ |\ \epsilon$$
$$F \rightarrow (\ E\ )\ |\ \texttt{id}$$

Step 1: Create the chart.

$$E \rightarrow T\ E'$$
$$E' \rightarrow +\ T\ E'\ |\epsilon$$
$$T \rightarrow F\ T'$$
$$T' \rightarrow *\ F\ T'\ |\ \epsilon$$
$$F \rightarrow (\ E\ )\ |\ \texttt{id}$$

E ={}
E'={}
T ={}
T'={}
F ={}

First Set Answer

Step 2: Add terminals and epsilons.

$$E \rightarrow T\ E'$$
$$E' \rightarrow +\ T\ E'\ |\epsilon$$
$$T \rightarrow F\ T'$$
$$T' \rightarrow *\ F\ T'\ |\ \epsilon$$
$$F \rightarrow (\ E\ )\ |\ \texttt{id}$$

E ={}
E'={+, ϵ}
T ={}
T'={*, ϵ}
F ={(, id}

Step 3: T starts with F; so add F entries to T.

$$E \rightarrow T\ E'$$
$$E' \rightarrow +\ T\ E'\ |\epsilon$$
$$T \rightarrow F\ T'$$
$$T' \rightarrow *\ F\ T'\ |\ \epsilon$$
$$F \rightarrow (\ E\ )\ |\ \texttt{id}$$

E ={}
E'={+, ϵ}
T ={(, `id`}
T'={*, ϵ}
F ={(, `id`}

Step 4: E starts with T; so add T entries to E.

$E \rightarrow T\ E'$	E ={(, `id`}
$E' \rightarrow +\ T\ E'\ \vert \epsilon$	E'={+, ϵ}
$T \rightarrow F\ T'$	T ={(, `id`}
$T' \rightarrow *\ F\ T'\ \vert\ \epsilon$	T'={*, ϵ}
$F \rightarrow (\ E\ )\ \vert\ \mathtt{id}$	F ={(, `id`}

First Set Answer

$$E \longrightarrow T\ E'$$
$$E' \longrightarrow +\ T\ E'\ |\epsilon$$
$$T \longrightarrow F\ T'$$
$$T' \longrightarrow *\ F\ T'\ |\ \epsilon$$
$$F \longrightarrow (\ E\ )\ |\ \mathtt{id}$$

E ={(, `id`}

E'={+, ϵ}

T ={(, `id`}

T'={*, ϵ}

F ={(, `id`}

Create a chart, add \$ to S.

$E \rightarrow T\ E'$	E ={ \$}
$E' \rightarrow +\ T\ E'\ \vert \epsilon$	E'={}
$T \rightarrow F\ T'$	T ={}
$T' \rightarrow *\ F\ T'\ \vert\ \epsilon$	T'={}
$F \rightarrow (\ E\ )\ \vert\ \texttt{id}$	F ={}

Add) to E.

$E \longrightarrow T\ E'$	E ={), $ }
$E' \longrightarrow +\ T\ E'\ \vert \epsilon$	E'={}
$T \longrightarrow F\ T'$	T ={}
$T' \longrightarrow *\ F\ T'\ \vert\ \epsilon$	T'={}
$F \longrightarrow (\ E\ )\ \vert\ \mathtt{id}$	F ={}

Follow Set Answer

Add E to E'.

$E \to T\ E'$	E ={), $ }
$E' \to +\ T\ E' \mid \epsilon$	E'={), $ }
$T \to F\ T'$	T ={}
$T' \to *\ F\ T' \mid \epsilon$	T'={}
$F \to (\ E\ ) \mid \texttt{id}$	F ={}

E' could be ϵ, so Add E to T.

$E \rightarrow T\ E'$	E ={), $ }
$E' \rightarrow +\ T\ E'\ \vert \epsilon$	E'={), $ }
$T \rightarrow F\ T'$	T ={), $ }
$T' \rightarrow *\ F\ T'\ \vert\ \epsilon$	T'={}
$F \rightarrow (\ E\ )\ \vert\ \mathtt{id}$	F ={}

Follow Set Answer

E' follows T, so add $First(E')$ to T.

$E \rightarrow T\ E'$	E ={), $ }
$E' \rightarrow +\ T\ E'\ \vert \epsilon$	E'={), $ }
$T \rightarrow F\ T'$	T ={+,), $ }
$T' \rightarrow *\ F\ T'\ \vert\ \epsilon$	T'={}
$F \rightarrow (\ E\ )\ \vert\ \texttt{id}$	F ={}

Follow Set Answer

T ends with T', so add T to T'.

$$E \rightarrow T\ E'$$
$$E' \rightarrow +\ T\ E' \mid \epsilon$$
$$T \rightarrow F\ T'$$
$$T' \rightarrow *\ F\ T' \mid \epsilon$$
$$F \rightarrow (\ E\ ) \mid \texttt{id}$$

E ={), $ }
E'={), $ }
T ={+,), $ }
T'={+,), $ }
F ={}

Follow Set Answer

T' could be ϵ, so add T to F.

$$E \rightarrow T\ E'$$
$$E' \rightarrow +\ T\ E'\ |\epsilon$$
$$T \rightarrow F\ T'$$
$$T' \rightarrow *\ F\ T'\ |\ \epsilon$$
$$F \rightarrow (\ E\ )\ |\ \texttt{id}$$

E ={), $ }
E'={), $ }
T ={+,), $ }
T'={+,), $ }
F ={+,), $ }

T' follows F, so add $First(T')$ to F.

$$E \rightarrow T\ E'$$
$$E' \rightarrow +\ T\ E'\ |\epsilon$$
$$T \rightarrow F\ T'$$
$$T' \rightarrow *\ F\ T'\ |\ \epsilon$$
$$F \rightarrow (\ E\ )\ |\ \texttt{id}$$

E ={), $ }
E'={), $ }
T ={+,), $ }
T'={+,), $ }
F ={+, *,), $ }
... and now we're done.